Statistical Research Skill

Group 6

School of Mathematics, University of Edinburgh

Model Structure

The Bayesian hierarchical spatio-temporal (BHST) model is specified as:

$$y_{s,t} = \mu + u_s + \delta(t) + \alpha_{\text{month}(t)} + \gamma_t + \epsilon_{s,t},$$

where:

- $y_{s,t}$ denotes the observed temperature at spatial location s and time t.

Model Components

- **Global mean (μ)**
  Represents the overall average temperature level across all spatial locations and time periods.
- **Spatial structured effect (u_s)**
  Models spatial dependence between neighbouring grid cells.
  This component is specified using a Conditional Autoregressive (CAR) or BYM2 prior, allowing geographically nearby locations to exhibit similar temperature patterns.
- **Smooth temporal trend ($\delta(t)$)**
  Captures long-term climate evolution over time.
  A second-order random walk (RW2) prior is used to allow a flexible, smooth nonlinear trend without imposing a strict linear structure.
- **Seasonal component ($\alpha_{\mathbf{month}(t)}$)**
  Represents the repeating annual cycle in temperature.
  This is modelled as a 12-level categorical effect corresponding to the month of the year, typically constrained to sum to zero for identifiability.
- **Temporal autocorrelation (γ_t)**
  Accounts for short-term persistence between consecutive observations.
  This component is modelled using an autoregressive process of order one (AR(1)).
- **Observation error ($\epsilon_{s,t}$)**
  Represents unexplained variability, assumed to follow:

$$\epsilon_{s,t} \sim \mathcal{N}(0, \sigma^2).$$

The model decomposes temperature variability into: spatial structure, long-term temporal trend, seasonal periodic effects, short-term temporal dependence, and residual noise.

```r
# ============================================================
# Model:
#   y_{s,t} = mu + u_s (BYM2) + delta(t) (RW2) + alpha_month(t) +
#             gamma_t (AR1) + eps
# ============================================================
# ---------- 0) Packages ----------
library(terra)
```

R Code

```
## Warning: package 'terra' was built under R version 4.5.2
```

```
## terra 1.8.93
```

```r
library(sf)
```

```
## Warning: package 'sf' was built under R version 4.5.2
```

```
## Linking to GEOS 3.13.1, GDAL 3.11.4, PROJ 9.7.0; sf_use_s2() is TRUE
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:terra':
##
##     intersect, union
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.5.2
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:terra':
##
##     extract
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:terra':
##
##     intersect, union

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(INLA)
```

```
## Warning: package 'INLA' was built under R version 4.5.2

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

##
```

```r
# ---------- 1) Read Scotland-only NetCDF from python process ----------
r_var <- rast("ds_scotland.nc", subds = "anom")  # loads only anomaly tmp data

# Inspect what's inside
print(r_var) # names(r_var): to see layer names
```

```
## class       : SpatRaster
## size        : 13, 17, 1488  (nrow, ncol, nlyr)
## resolution  : 0.5, 0.5  (x, y)
## extent      : -9, -0.5, 54.5, 61  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (CRS84) (OGC:CRS84)
## source      : ds_scotland.nc:anom
## varname     : anom
## names       : anom_1, anom_2, anom_3, anom_4, anom_5, anom_6, ...
## time (days) : 1901-01-16 to 2024-12-16 (1488 steps)
```

```r
# ---------- 2) Dates from NetCDF time metadata ----------
dates <- terra::time(r_var)
if (is.null(dates)) {
  stop("No time metadata found in ds_scotland.nc. Print names(r_var) and we can parse dates safely.")
}
```

```r
dates <- as.Date(dates)

layer_names <- names(r_var)
T <- nlyr(r_var)

# ---------- 3) Wide dataframe (lon/lat + layers) ----------
# na.rm=TRUE drops NA cells (outside Scotland) for each layer automatically
df_wide <- as.data.frame(r_var, xy = TRUE, na.rm = TRUE) |>
  rename(lon = x, lat = y)

# ---------- 4) Long dataframe (one row per cell-month) ----------
df_long <- df_wide |>
  pivot_longer(
    cols = -c(lon, lat),
    names_to = "layer",
    values_to = "temp"
  ) |>
  mutate(
    t = match(layer, layer_names),
    date = dates[t],
    year = year(date),
    month = month(date),
    cell = interaction(round(lon, 4), round(lat, 4), drop = TRUE)  # stable cell key
  ) |>
  select(lon, lat, cell, t, date, year, month, temp) |>
  filter(!is.na(temp))

cat("Rows N =", nrow(df_long), "\n")
```

```
## Rows N = 65472
```

```r
cat("Time points T =", length(unique(df_long$t)), "\n")
```

```
## Time points T = 1488
```

```r
cat("Cells S =", length(unique(df_long$cell)), "\n")
```

```
## Cells S = 44
```

```r
# ---------- 5) Build spatial cell map + rook adjacency ----------
# Round coordinates to avoid floating mismatches in indexing
xy_map <- df_long |>
  distinct(cell, lon, lat) |>
  mutate(
    lon_r = round(lon, 4),
    lat_r = round(lat, 4)
  )

xs <- sort(unique(xy_map$lon_r))
ys <- sort(unique(xy_map$lat_r))
```

```r
xy_map <- xy_map |>
  mutate(
    ix = match(lon_r, xs),
    iy = match(lat_r, ys)
  ) |>
  arrange(ix, iy) |>
  mutate(cell_id = row_number())

S <- nrow(xy_map)

# Lookup (ix,iy) -> cell_id
lookup <- setNames(xy_map$cell_id, paste(xy_map$ix, xy_map$iy, sep=","))

neighbors <- vector("list", S)
for (k in seq_len(S)) {
  ix <- xy_map$ix[k]
  iy <- xy_map$iy[k]
  cand <- c(paste(ix+1, iy, sep=","),
            paste(ix-1, iy, sep=","),
            paste(ix, iy+1, sep=","),
            paste(ix, iy-1, sep=","))
  nb <- lookup[cand]
  neighbors[[k]] <- as.integer(nb[!is.na(nb)])
}

cat("Zero-neighbour cells:", sum(sapply(neighbors, length) == 0), "\n")
```

```
## Zero-neighbour cells: 3
```

```r
cat("Neighbour count summary:\n")
```

```
## Neighbour count summary:
```

```r
print(summary(sapply(neighbors, length)))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.000   3.000   2.909   4.000   4.000
```

```r
# ---------- 6) Write INLA adjacency file manually + read graph ----------
adj_file <- tempfile(fileext = ".adj")
con <- file(adj_file, open = "w")

# First line: number of nodes
writeLines(as.character(S), con)

# Each subsequent line: node_id n_neighbors neighbor1 neighbor2 ...
for (i in seq_len(S)) {
  nb_i <- neighbors[[i]]
  line <- paste(
    i,
    length(nb_i),
```

```
    if (length(nb_i) > 0) paste(nb_i, collapse = " ") else ""
  )
  writeLines(line, con)
}
close(con)

g <- INLA::inla.read.graph(adj_file)

# ---------- 7) Prepare INLA modelling data ----------
df_inla <- df_long |>
  left_join(xy_map |> select(cell, cell_id), by = "cell") |>
  mutate(
    month_id = month,  # 1..12
    time_rw  = t,      # RW2 index
    time_ar  = t       # AR(1) index
  )

# ---------- 8) Fit BHST model in INLA ----------
formula <- temp ~ 1 +
  f(month_id, model = "iid", constr = TRUE) +     # seasonality (sum-to-zero)
  f(time_rw,  model = "rw2", constr = TRUE) +     # smooth long-term trend
  f(time_ar,  model = "ar1") +                    # short-term persistence
  f(cell_id,  model = "bym2", graph = g)          # spatial structured + unstructured

fit <- inla(
  formula,
  family = "gaussian",
  data = df_inla,
  control.fixed = list(
    mean = list("(Intercept)" = 0),
    prec = list("(Intercept)" = 1/100)   # variance = 100 = 10^2
  ),
  control.predictor = list(compute = TRUE),
  control.compute = list(dic = TRUE, waic = TRUE, cpo = TRUE)
)

print(summary(fit))
```

```
## Time used:
##     Pre = 18.3, Running = 32, Post = 1.26, Total = 51.6
## Fixed effects:
##             mean    sd 0.025quant 0.5quant 0.975quant  mode kld
## (Intercept) 0.214 0.034      0.147    0.214      0.281 0.214   0
##
## Random effects:
##   Name     Model
##     month_id IID model
##    time_rw RW2 model
##    time_ar AR1 model
##    cell_id BYM2 model
##
## Model hyperparameters:
##                                          mean       sd 0.025quant 0.5quant
```

```
## Precision for the Gaussian observations 3.90e+01 2.18e-01   3.86e+01 3.90e+01
## Precision for month_id                  8.16e+02 3.30e+03   4.47e+01 1.93e+02
## Precision for time_rw                   1.70e+05 5.90e+04   8.62e+04 1.59e+05
## Precision for time_ar                   9.70e-01 3.90e-02   8.96e-01 9.69e-01
## Rho for time_ar                         2.50e-01 2.60e-02   1.97e-01 2.51e-01
## Precision for cell_id                   3.08e+03 8.45e+02   1.61e+03 3.02e+03
## Phi for cell_id                         5.22e-01 1.39e-01   2.62e-01 5.20e-01
##                                         0.975quant     mode
## Precision for the Gaussian observations   3.95e+01 3.90e+01
## Precision for month_id                    5.38e+03 5.88e+01
## Precision for time_rw                     3.15e+05 1.40e+05
## Precision for time_ar                     1.05e+00 9.67e-01
## Rho for time_ar                           3.00e-01 2.52e-01
## Precision for cell_id                     4.89e+03 2.93e+03
## Phi for cell_id                           7.93e-01 4.98e-01
##
## Deviance Information Criterion (DIC) ...............: -52586.73
## Deviance Information Criterion (DIC, saturated) ....: 66999.15
## Effective number of parameters .....................: 1527.62
##
## Watanabe-Akaike information criterion (WAIC) ...: -52569.90
## Effective number of parameters .................: 1510.14
##
## Marginal log-Likelihood:  21409.80
## CPO, PIT is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

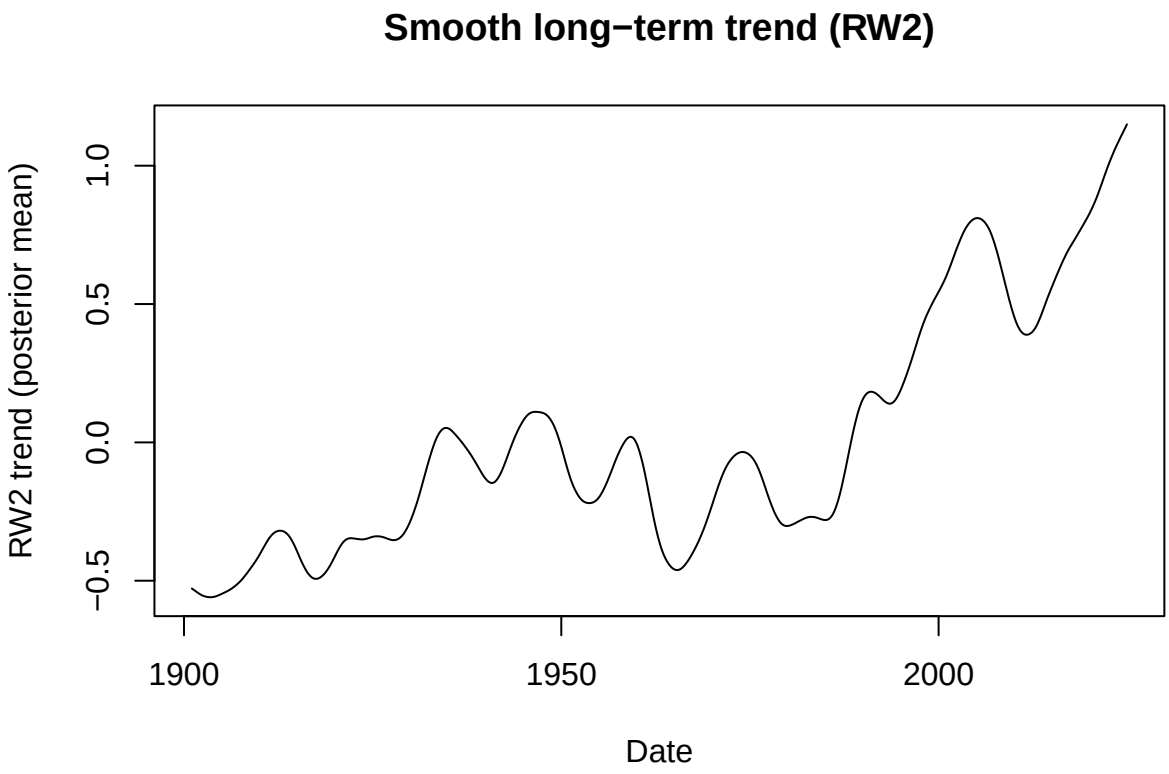
```
cat("DIC:", fit$dic$dic, "\n")
```

```
## DIC: -52586.73
```

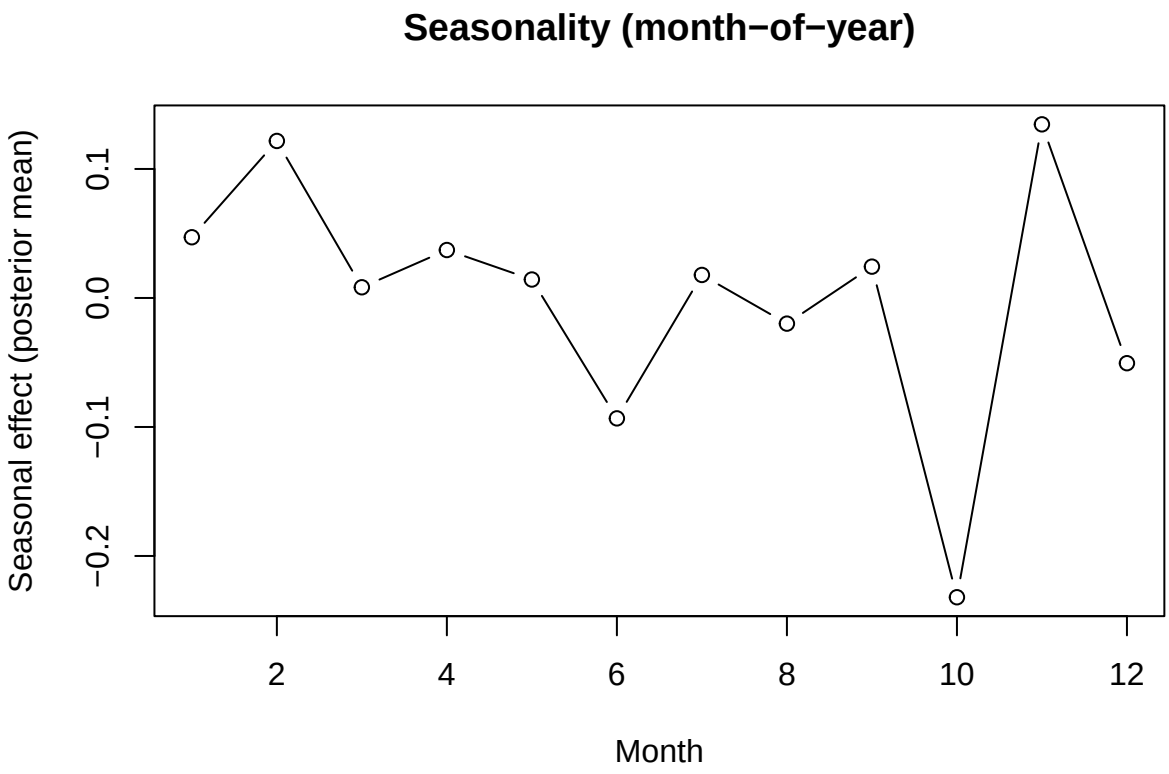
```
cat("WAIC:", fit$waic$waic, "\n")
```

```
## WAIC: -52569.9
```

```
# ---------- 9) Plot posterior mean components ----------
# RW2 trend
trend_post <- fit$summary.random$time_rw
plot(dates, trend_post$mean, type = "l",
     xlab = "Date", ylab = "RW2 trend (posterior mean)",
     main = "Smooth long-term trend (RW2)")
```




```r
# Seasonality (month effects)
seas_post <- fit$summary.random$month_id
plot(1:12, seas_post$mean, type = "b",
     xlab = "Month", ylab = "Seasonal effect (posterior mean)",
     main = "Seasonality (month-of-year)")
```

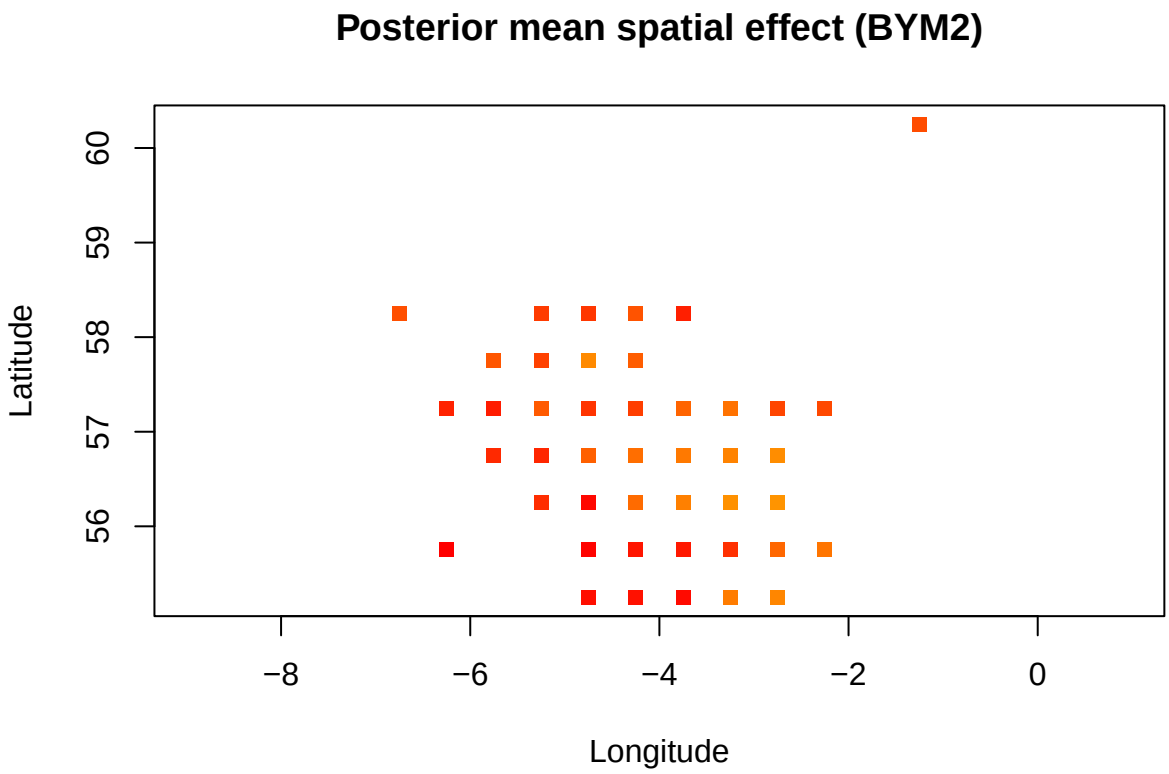



```
# Spatial effect (BYM2): plot on lon/lat points
sp_post <- fit$summary.random$cell_id

sp_df <- data.frame(
  cell_id = as.integer(rownames(sp_post)),
  spatial_mean = sp_post$mean
)

xy_plot <- xy_map %>%
  left_join(sp_df, by = "cell_id")

cols <- heat.colors(100)
plot(xy_plot$lon, xy_plot$lat,
     col = cols[rank(xy_plot$spatial_mean, ties.method="first")],
     pch = 15, asp = 1,
     xlab = "Longitude", ylab = "Latitude",
     main = "Posterior mean spatial effect (BYM2)")
```


Posterior mean spatial effect (BYM2)
Latitude
60
59
58
57
56
−8
−6
−4
−2
0
Longitude